Assignment 1 Solution

1. Informed Search (4 points)

1(a) (1 point)

Consider H_1 = *Misplaced Tile Heuristic* and H_2 = Manhattan Distance *Heuristic*. Acceptable "proofs" should have that H_2 is better than H_1 because H_2 is closer to the correct cost, *i.e.*, $H^*(n) \geq H_2(n) \geq H_1(n)$, where $H^*(n)$ is the real cost. This implies that the time taken to do search should be lower. In other words, H_2 provides a tighter lower bound.

1(b) (1 point)

For a heuristic to be consistent, it must comply with the following:

$$h(n) \leq c(n,n') + h(n') \ \forall n,n'$$

Proposition: H_1 is consistent

Proof: 3 cases need to be addressed:

1 - After a move, the SAME NUMBER of tiles is out of position:

In this case, $c(n,n') = 1$ and $H(n) = H(n')$. So, $h(n) \leq c(n,n') + h(n')$.

2 - After a move, one LESS tile is out of position:

In this case, $c(n,n') = 1$ and $H(n) = H(n') + 1$. So, $h(n) \leq c(n,n') + h(n')$.

3 - After a move, one MORE tile is out of position:

In this case, $c(n,n') = 1$ and $H(n) = H(n') - 1$. So, $h(n) \leq c(n,n') + h(n')$.

These 3 cases show that no matter the move, we will have $h(n) \leq c(n,n') + h(n')$, which means that after each move the *f-cost* increases or stays the same. Therefore, the *f-cost* increases monotonically.

Proposition: H_2 is consistent

The proof for this proposition is almost identical to the proof given above, with the exception that the first case is not possible anymore. When the blank moves, H_2 's value is either *-1* or *+1* from *n* to *n'*. Note: An answer like this one was given full marks.

2(a) (1 point)

Time complexity: $O(b^d)$; in the worst case it may still take exponential time since all nodes up to a certain f limit need to be expanded, which takes exponential time with respect the depth of the shallowest solution.

Space complexity: $O(bd)$: the algorithm needs to store all nodes in a path with all children (at most b) of each node on the path.

For full marks, the complexity in big O notation is required (as opposed to simply stating exponential and linear complexity).

2(b) (0.5 point)

*IDA** is complete because the algorithm uses a *DFS* scheme to visit exhaustively the tree of solutions while increasing continually the fringe depth limit after each iteration, until the goal is reached. This limit stops the search from going down the tree indefinitely. As a result admissibility and consistency are **not** needed for completeness.

Note 1: The fact that the fringe ceiling is derived from an *f-cost* value doesn't change this assumption as long as the branching factor is finite, and that the costs are positive (greater than some constant).

Note 2: This is only true if we assume the computer has enough resources available (memory).

2(c) (0.5 point)

*IDA** is optimal because it uses a *DFS* scheme similar to *IDS*, and *IDS* is optimal. Every time *IDA** chooses a new *cutoff* value, it chooses a value that is larger than its previous *cutoff*, but that is the smallest value among all the *f-cost* values encountered so far. Hence, the optimality relies upon the consistency of the heuristic in use, which guarantees that the *f-costs* are monotonically increasing by some 'optimistic' amount. Note that admissibility is not sufficient for graphs. We need consistency of the heuristic to ensure optimality of IDA*.

2. Constraint Satisfaction (6 points)

1. (1 point)

Example of a possible solution:

Variables: V_1 ,...,V_{81}
Domain: $\{1,...,9\}$
Constraints:

- Each row contains integers from 1 to 9 without duplications.

- Each column contains integers from 1 to 9 without duplications.
- Each box contains integers from 1 to 9 without duplications.

2. (5 points)

Example of an acceptable solution for time and number of steps (nodes):

Nodes

	BT		BT+FC		BT+FC+H	
	Mean	SD	Mean	SD	Mean	SD
Easy	5,111,648	2,946,634	1,638	1,205	45	0
Medium	12,683,351	7,839,447	6,409	4,342	112	13
Hard	1,121,986,866	689,324,487	232,723	158,168	111	20
Evil	>1,826,400,000		1,698,605	1,116,862	66	13

				Time (seconds)		
	BT		BT+FC		BT+FC+H	
	Mean	SD	Mean	SD	Mean	SD
Easy	20	11	0.050	0.037	0.0052	0.0009
Medium	50	31	0.199	0.135	0.0149	0.0097
Hard	4,563	2,798	6.916	4.670	0.0111	0.0029
Evil	>7,200		50.489	33.283	0.0064	0.0015

Note: Clearly, the running times depend on the implementation. Those results are based on a Python implementation where averages and standard deviations are computed based on a sample of 50 runs. Since BT takes too long for Evil, the running time was cut off at 2 hours.

Explanation of the results: As expected, BT+FC+H expands fewer nodes and takes less time than BT+FC. Similarly, BT+FC expands fewer nodes and takes less time than BT. Note however, that the time per expanded node is greater for BT+FC+H than BT+FC due to the overhead introduced by the heuristics. Similarly, the time per expanded node is greater for BT+FC than BT due to the overhead introduced by forward checking. The results clearly show a dramatic reduction in time and expanded nodes as forward checking and the heuristics are incorporated.